Bolt Beranek and Newman Inc.

Report No. 5492

Combined Quarterly Technical Report No. 31

Pluribus Satellite IMP Development
Mobile Access Terminal Network

November 1983

Prepared for:
Defense Advanced Research Projects Agency

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A136 056	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Combined Quarterly Technical Report No. 31		5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical 9/1/83 to 11/30/83
		6. PERFORMING ORG. REPORT NUMBER 5492
7. AUTHOR(s) S. Blumenthal		8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0353 N00039-81-C-0408
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Arpa Order No. 3214
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE December 1983 13. NUMBER OF PAGES 32
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DSSW Room ID The Pentagon Washington, DC 20310 NAVALEX Washington, DC 20360		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE/DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer networks, packets, packet broadcast, satellite communication, gateways, UNIX, Pluribus Satellite IMP, shipboard communications, ARPANET, Internet.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Quarterly Technical Report describes work on the development of Pluribus Satellite IMPs; and on shipboard satellite communications.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

BLANK PAGE

Report No. 5492

COMBINED QUARTERLY TECHNICAL REPORT NO. 31

PLURIBUS SATELLITE IMP DEVELOPMENT
MOBILE ACCESS TERMINAL NETWORK

November 1983

This research was supported by the Defense Advanced Research Projects Agency under the following contracts:

MDA903-80-C-0353, ARPA Order No. 3214
N00039-81-C-0408

Submitted to:

Director
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Attention: Program Management

Table of Contents

1 INTRODUCTION

This Quarterly Technical Report is the current edition in a series of reports which describe the work being performed at BBN in fulfillment of several ARPA work statements. This QTR covers work on several ARPA-sponsored projects including (1) development of the Pluribus Satellite IMP; and (2) development of the Mobile Access Terminal Network. This work is described in this single Quarterly Technical Report with the permission of the Defense Advanced Research Projects Agency. Some of this work is a continuation of efforts previously reported on under contracts DAHC15-69-C-0179, F08606-73-C-0027, F08606-75-C-0032, MDA903-76-C-0214, MDA903-76-C-0252, N00039-79-C-0386, and N00039-78-C-0405.

2 PLURIBUS SATELLITE IMP DEVELOPMENT

2.1 Introduction

Progress continued to be made during the quarter toward stable operation of multiple sites at 3.088Mb/s. In this QTR, we report on current Wideband Network status, operations, and performances. In addition, we report on progress in BSAT.

Progress on BSAT software development concentrated on two areas. Code to handle the scheduling and managing of channel streams was written but has not yet been debugged, and a satellite channel simulator for the BSAT based on a software Butterfly processor node was designed. A detailed description of the BSAT Satellite Simulator design is given in Section 2.5.

2.2 Wideband Network Status

One of the major goals for this quarter was to increase the number of operational sites in the network. Toward this goal, the SRI ESI was installed at RADC during October, and the Wideband Network gained a third operational site on the 28th, after ice damage to the earth station's underground waveguide was repaired.

The earth station installation and check-out was completed for both Forts Huachuca and Monmouth during the quarter. The Fort Monmouth PSAT was installed 19 September. After installation, it was left powered down because of a lack of proper air conditioning. The Fort Huachuca PSAT has also been left powered down because of a lack of proper air conditioning.

Testing of the production model of the ESI (the ESI-A) took place at ISI during much of the quarter and was completed in late October. Production is scheduled to begin in mid-November.

A new version of the PSAT software was released on 5 October (release 1.003.00). Many bugs affecting overall system reliability present in previous releases were eliminated, and several new features were added. Among these were collection of T&M data, an increase in buffer pool size, and facilities for simplified NOC monitoring and control.

2.3 Wideband Network Operations

During the quarter, the Wideband Network operated consistently and reliably in a two-site configuration composed of the ISI and Lincoln sites. Of all the Wideband sites, ISI and Lincoln were the only ones with sufficient equipment for channel operation; most of the remaining sites were lacking ESI's.

Neither ISI nor Lincoln experienced any serious problems during the quarter. Most of the difficulties encountered involved small-scale hardware failures in the PSAT. These were often remedied by adjusting configurations to include redundant components. Other failures not covered by redundant components were repaired within one day by BBN field service.

At the other sites, the PSATs generally ran well. There was a power failure at RADC in early August that damaged the PSAT. The air conditioning system failed to restart after the power failure, and the resulting temperature rise caused overheating in the PSAT. After being repaired, the PSAT was kept powered down until a thermal cutoff mechanism could be installed.

ARPANET/MILNET split tests occurred on 9-10 September and 15-16 September. During these days, BBN experimented with alternate site reload and monitoring configurations. Software to load and debug the PSATs using the satellite NU system running on BBN-INOC was debugged, completing the transition from TENEX-based to UNIX-based monitoring and control.

2.4 Wideband Network Performance

On August 1, the Wideband Network task force reported back to DARPA and DCA. They reported that much progress had been

made, but that reliable 3 Mb/s service involving all Wideband Network sites had not yet been achieved. An agenda was established for continued task force activity. The following items were identified as near-term tasks: begin testing two sites at 3 Mb/s, continue host-level testing of the network, evolve measurement and reporting tools, evolve methods of network and subsystem configuration control, begin collecting and using T&M data to monitor network performance, and begin testing CPODA.

During August the network operated stably at 1.5 Mb/s, with some testing done at 3 Mb/s. An extensive set of tests at 3 Mb/s was conducted during the August 22 task force visit to ISI, and it was determined that a two-site network operated reliably at 3 Mb/s with the channel control information coded at rate 1/2. A significant number of bit errors in the data were encountered when the speech was uncoded (rate 1). It was found that these bit errors could be significantly reduced if the speech was coded at rate 3/4.

Lincoln Labs and ISI cooperated in channel throughput performance tests on 29-30 September. Two simultaneous telephone-to-telephone connections were sustained, in addition to other looped voice connections from Lincoln. These tests represented the highest voice traffic load yet carried by the Wideband Satellite Network.

2.5 BSAT Satellite Simulator

This section describes the design of a software-based satellite channel simulator for the BSAT. The primary function of the Satellite Simulator is to connect multiple BSATs into a network via a simulated satellite channel. In this role it must provide the approximately 1/4-second propagation delay that would be expected over a channel supplied by a geostationary satellite such as WESTAR. It must also simulate the multiaccess/broadcast capability of a satellite channel. This requires merging the bursts of data transmitted by different BSATs so that any transmissions that arrive at an actual satellite at the same time result in corrupted data on the simulated satellite downlink. The merged satellite downlink data must be received by all of the connected BSATs. In addition to the channel, this simulator will simulate the functions of the Earth Station Interface (ESI) (modem/codec) supplied by Linkabit.

Satellite simulators performing the primary function described above have been built out of dedicated digital hardware. The burst merging function was provided by logically ORing any bits that were clocked into different input ports of the simulator at the same time. The resulting merged data were then delayed for one Round Trip Time (RTT) before being clocked out of each of the output ports. Such simulators have proved

themselves quite useful for the development and debugging of those components of Satellite IMP software that can only be properly exercised in the presence of actual satellite delay or in a multi-node network environment.

The BSAT Satellite Simulator will be different from the satellite simulators described above in that it will be implemented as application software running under the control of the Chrysalis operating system on a Butterfly Multiprocessor. Such an implementation has been chosen for compatibility with the current BSAT. The latter also runs in the Chrysalis/Butterfly environment and expects to access the satellite channel via an ESI connected to one or more of the serial synchronous I/O channels provided by the standard Butterfly I/O (BIO) board. The ESI is expected to be an intelligent device providing such signal processing functions as data encoding/decoding, modulation/demodulation, etc. under the control of the BSAT.

With a software realization of a satellite channel simulator, it is easier to simulate the behavior of the actual ESI to which the BSAT would be attached. This includes the exchange of local control packets between the BSAT and the ESI (used by the BSAT to read/modify ESI parameters, read the value of the ESI's local time clock, set loopback modes, reset the ESI, etc.) and discarding (before delivery to the BSATs) any burst

whose preamble or ESI control area was corrupted by a collision at the satellite. Additionally, it is easy for a software simulator to keep a variety of useful statistics on the traffic traversing the satellite channel and the BSAT-ESI interfaces. Such functions are included in the current design.

A software implementation also makes it easier to simulate such satellite channel characteristics as uplink noise, downlink noise, RTT drifts, etc. These functions, and other possible test functions, will not be implemented in the initial Simulator; however, it is hoped that the current design will facilitate their possible insertion at a later date.

2.5.1 Model of Simulator Operation

2.5.1.1 Delay Function

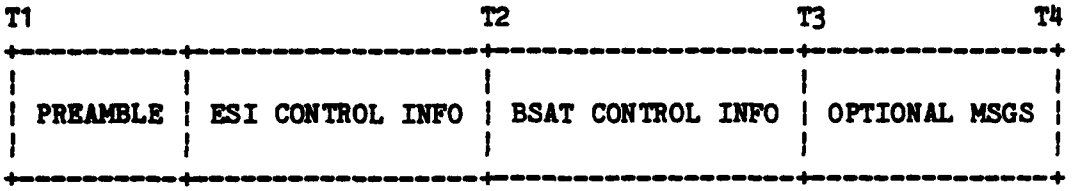
The burst packets submitted by the BSATs to the ESI for transmission over the satellite channel will be stamped with a 32-bit "local" time. This time is in units of 1/3.088 x 10**-6 sec. A clock is expected to run at this rate in the ESI and the time stamp indicates the clock's value for the burst preamble tranmission time. Bursts are passed to the ESI up to two PODA frames before the indicated local transmit time (~40 msec). When the ESI acquires a burst from the satellite downlink, it prepends

the burst preamble's receive time (again in units of the ESI's local time clock) to the burst data before passing it to the BSAT. All bursts are received at the BSAT sites after an appropriate satellite propagation delay.

The BSAT Satellite Simulator will duplicate this function as closely as possible. Different RTTs for the different connected BSATs will be supported by performing the burst merging function with times normalized to "satellite" time and delaying the merged (satellite downlink) data different amounts of time for each BSAT. There will be tolerance parameters for both how far before, and how far after, an indicated local transmit time a burst may be received from a BSAT for transmission by the Simulator. (BIO latencies may cause the burst to appear late.)

2.5.1.2 Burst Merger and Reception

A transmitted burst is modeled in the Simulator as follows:



The PREAMBLE is used by the ESI to acquire the burst. The Simulator will drop (will not return to ANY of the BSATs) any burst whose PREAMBLE's satellite time coincides with transmissions from any other BSAT. It is also assumed that an ESI cannot acquire bursts transmitted at symbol rates different than that specified by the ESI's receive symbol rate parameter. The Satellite Simulator's downlink process for a given connected BSAT will therefore drop any bursts transmitted with symbol rates different than the BSAT's receive symbol rate.

The ESI CONTROL INFO is used by the ESI to determine the type of demodulation and decoding to be performed on the remainder of the burst before it is passed to the BSAT. It is protected by a checksum field inserted by the ESI on transmission and checked on reception. It is assumed that the ESI will drop any bursts with bad checksums; hence the Simulator will drop any burst whose ESI CONTROL INFO's satellite time coincides with

transmissions from any other BSAT. A further assumption is that a receiving ESI must have the same "state-1" modulation type and coding rate as a transmitting ESI in order to decode a burst properly; hence the Simulator's downlink process for a given connected BSAT will drop any bursts transmitted with non-matching state-1 parameters.

The BSAT CONTROL INFO is used by the BSAT for channel protocol operations. It is transparent to the ESI and is protected by a checksum field inserted and checked by the BSAT. If the transmissions from any other BSAT coincide with this part of a burst (and the burst's PREAMBLE and ESI CONTROL INFO are clear) the Simulator will return the burst packet to the BSATs, but with corrupted bits in the BSAT CONTROL INFO area. (This presumably will cause the BSATs to detect a checksum error and discard the burst packet.)

The OPTIONAL MSGS are the host messages and message headers being transmitted by the BSAT. They are transparent to the ESI, and may or may not be protected by various checksums. This burst section is handled by the Simulator in the same fashion as the BSAT CONTROL INFO.

The Simulator's burst merging algorithm operates on time values normalized to a 3.088 MHz "satellite" clock. The satellite times for the different parts of a burst are indicated

as {T1,T2,T3,T4} in the diagram above. T1 is the time of PREAMBLE start derived by the Simulator from the burst's transmission local time stamp and the BSAT's current RTT. T4 is derived from T1 and the BSAT-supplied "burst length" and "symbol rate" fields. The derivations of T2 and T3 additionally require Simulator parameters that may be a function of the state-1 modulation type and coding rate, as well as the exact burst format being used by the BSAT.

A collision will be deemed to have occurred, and the appropriate action (as described above) will be taken, when any part of a given burst area is in contention with another BSAT's burst. Data corruption will probably be done by complementing a few selected words in the given burst area.

2.5.1.3 Local Control Packet Exchanges

The Simulator will emulate ESI responses to BSAT commands to set/read its operational parameters. Some of the parameters, such as preamble length, will have an effect on Simulator operation; other parameters will simply be stored. A BSAT will also be able to reset and loop its Simulator port. The REQUEST LOCAL TIME and LOCAL TIME REPLY packet exchanges needed by BSATs to derive global time from their Butterfly processor clocks will be supported. If the BSAT generates ACQUIRE GROSS FREQUENCY

OFFSET, BUILT IN TEST, or similar packets, canned responses will be returned by the Simulator. ILLEGAL LOCAL CONTROL PACKET RECEIVED and PROTOCOL ERROR packets may also be generated by the Simulator. T&M RESULTS will not be supported.

The Simulator software is structured to provide

a. independent control of each BSAT interface for looping, resetting, and parameter manipulation operations, and

b. transmission of local control packets to the BSATs as quickly as possible. (They will be pushed to the front of the queue of items to be sent to a BSAT and will not be subject to any propagation delays.)

2.5.2 Simulator Hardware Configuration

The Satellite Simulator will communicate with connected BSATs via the serial synchronous I/O channels provided by the currently available BIO boards. Note, however, that most of the Simulator software will be independent of the specific Butterfly I/O boards being used, so that it will not be difficult to adapt the Simulator to new boards as they become available.

The Simulator (and the BSATs) must deal with the current bandwidth limitations of the BIO channels, the BIO microprocessor, and the Butterfly I/O bus (the BIOLINK). Two of these limitations are the following:

- Each channel has a limit of 2 Mbps in each direction.
- The BIO microprocessor can handle an instantaneous data rate of up to about 5 Mbps.

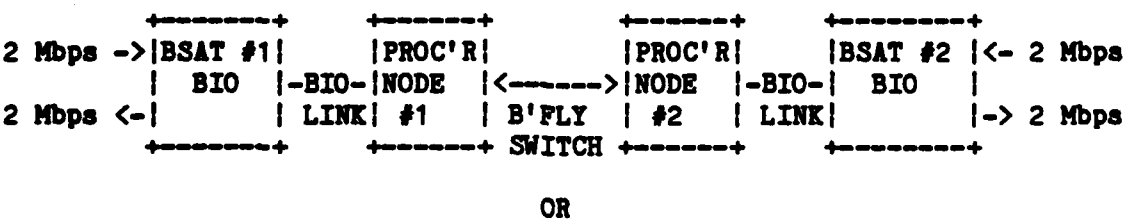
These numbers are significant in light of the fact that the Wideband network runs at satellite channel data rates of up to 3.088 Mbps. If such high-rate operation is to be supported, it is necessary to use two ports for each BSAT-to-ESI connection and two ports for each ESI-to-BSAT connection. The second limitation above permits only two of these ports to be located on each BIO board; each complete BSAT-ESI interface would therefore require two BIO boards for 3.088 Mbps operation, and a BSAT Satellite Simulator connected to two BSATs would require four BIO boards (not to mention the other four BIO boards required by the two BSATs).

To keep the number of needed BIO boards down and to simplify the initial BSAT-Simulator integration, each BSAT-Simulator connection will initially use a single 2 Mbps channel in each direction. In order to accommodate the current BSAT implementation, this will be done with two half-duplex ports rather than a single full-duplex port.

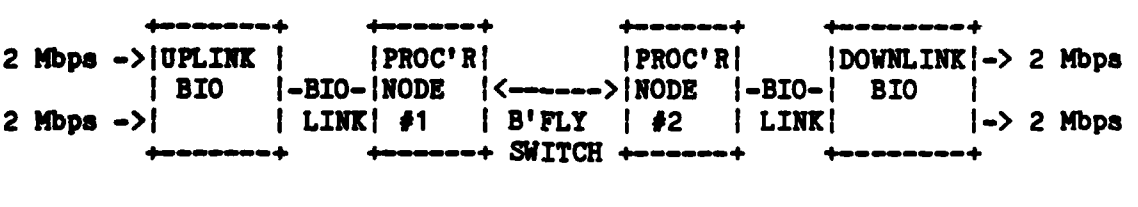
Another issue is the number of Butterfly processor nodes that will be needed to support the Simulator. The Satellite Simulator will most often interface to only one or two BSATs

(although nothing in the structure of its software will prevent it from handling a greater number of BSATs). As discussed above, a two-BSAT configuration will initially require two BIO boards. These boards could conceivably be connected to a single processor node. This would have the advantage of allowing the Satellite Simulator to live in the same Butterfly machine as a BSAT without perturbing the BSAT's operation, since the all of the Simulator software would reside in the single node and would not do any accesses across the Butterfly Switch. This consideration aside, it is probably best to support a two-BSAT Simulator with a minimum of two processor nodes, each interfaced to a BIO board. This configuration has the advantages of not requiring the movement of BIO boards from their standard lab configuration when the Simulator is to be run, and it provides the Simulator with more physical memory for its satellite delay function.

A two-BSAT Simulator will therefore look like:



OR



The process structure of the Simulator will be able to handle either of these configurations. Which configuration to use will be determined by the performance observed during BSAT-Simulator testing.

2.5.3 Simulator Software

Like the BSAT software, the Satellite Simulator software will be implemented as a set of C-language-coded processes running under the control of the Chrysalis operating system. Some of the BSAT program's functions will be employed for use by the Simulator where this seems efficient. The Simulator software will be structured so as to be independent of the number of connected BSATs, the number of physical ports used to connect each of the BSATs, and the location of the physical ports. Five kinds of processes make up the Simulator application:

a. Synchronous I/O driver processes (receive-only or transmit-only).

b. Uplink processes.

c. Downlink processes.

d. One merge, or satellite process.

e. One system initialization/control process.

The processes are described in the following sections.

2.5.3.1 Synchronous I/O Driver Processes

There will be one of these processes for every physical port that is connected to a BSAT. Each process will operate in either a transmit-only or receive-only mode, since the BSAT software does not use full-duplex ESI connections. Both the transmit-only and receive-only processes will be linked together and loaded as a single Chrysalis process template; one of the arguments passed to each separate invocation via Make_Process will be used to inform the invoked process of its direction. This is like the "process group" notion used in the BSAT software; it is intended to minimize the amount of physical memory used for process text segments when a number of different process types use many common routines.

All of the management and much of the initialization of CCBs, I/O device control registers, the transmit and receive queues used by the BIO microprocessor, etc. will be performed by routines developed for the BSAT. These routines will be changed as little as possible. The process initialization, startup, and event dispatching code, as well as the routines providing the interface to the uplink and downlink processes, will be unique to the Simulator application.

Since an I/O driver process must access the I/O device control registers for the physical port it is managing, it must

run on the processor node to which the given port is attached. It will map in all of the buffer pools located on the processor node.

A receive-only I/O driver process interfaces to a single uplink process through two dual queues: a receive-request queue and a task queue. The receive-request queue is used to pass packets received on the physical link to the uplink process, and the task queue is used by the uplink process to control the synchronous channel's modes/parameters. Packets suffering overruns or aborts will not be passed through to the uplink process; however, packets with CRC errors will be passed on to higher-level processing. An extra header word will be added to each packet indicating the total number of bytes in all of the buffers making up the packet.

A transmit-only I/O driver process interfaces to a single downlink process through two dual queues: a transmit-request queue and a task queue. The transmit-request queue is used to pass packets from the downlink process to the I/O driver for transmission over the physical link, and the task queue is used by the downlink process to control the synchronous channel's modes/parameters. The I/O driver process splices the packets found on the transmit-request queue into the BIO transmitter's CCB queue in accordance with the values in the timestamp field of

the buffer headers. This timestamp value determines the earliest time of transmission of the packet to the BSAT; it is calculated by the higher-level uplink, downlink, and merge processes.

2.5.3.2 Uplink Processes

There will be one of these processes for every receive-only I/O driver process. If each BSAT is connected to the Simulator by two physical links, there will be two receive-only I/O drivers and two uplink processes per BSAT. Each uplink process communicates with its corresponding I/O driver via a (private) pair of dual queues as previously described. In addition, each uplink process communicates with:

a. the merge process via a private uplink queue containing burst packets to be sent over the simulated satellite channel,

b. a downlink process via a downlink dual queue containing local control packets and burst packets for transmission to the BSAT with which the uplink process is associated, and

c. the same downlink process via a common memory segment.

There will be a separate uplink queue interfacing each uplink process to the merge process. The duplication of the processes/queues handling uplink data for multi-ported BSATs allows flexibility in the placement of the physical ports handling BSAT input. Also, since the merge process must handle

the time ordering of bursts from different BSATs, it is simplest to let it also handle the ordering of the bursts from a multi-ported BSAT.

An uplink process acts as a filter, passing non-looped burst packets to the satellite (the merge process) via its uplink queue, and looped burst packets and local control packet responses back to the BSAT via a downlink queue. An uplink process will not (except for processing/scheduling latencies) delay or re-order burst packets; this allows local control packets to be serviced quickly. It is assumed that the burst packets arrive at the Simulator in transmit-time order. (The merge process discovers out-of-order packets.)

Most of the satellite propagation delay to be suffered by burst packets will be waiting on an uplink queue for service by the merge process. Given current estimates of both buffer size and the waiting time on this queue, a ring-buffer-oriented dual queue will probably be usable. However, a linked list with appropriate interprocess dual-queue locks and events could be used if memory space is tight.

When a non-looped uplink process receives a burst packet, it derives the BSAT Simulator local time of the packet's arrival from a time stamp placed in the buffer header by the I/O receive process and does a validity check comparing the resulting value

with the packet's specified local ESI transmission time. The transmit time field of the packet is then incremented by 1/2 of the RTT value (in 3.088 MHz local time units) for the BSAT in question, normalizing it to a common satellite time for the merge process. The buffer header time-stamp field is adjusted to be consistent with the local transmission time and is then incremented by 1/2 RTT + K (in 16 KHz Butterfly time units). K is a Simulator parameter whose optimum value must be determined. Since the buffer header time-stamp field is used by the merge process to determine when to process the burst packet and send it to the downlink, larger values of K will reduce the buffering demands on the Simulator by forcing necessary downlink copying operations to occur closer to the time of burst packet delivery to the BSAT. If K is too large, however, bursts may arrive at the BSATs too late.

An uplink process will insert information needed by the merge and downlink processes in an extra buffer header word (e.g., the burst's state-1 modulation type and code rate, preamble size, an indication that the burst went via satellite rather than being looped). The buffers will then be placed on the appropriate uplink queue.

Most of the processing of a BSAT's local control packets is done by an uplink process. If the processing results in a

response that needs to be sent to the BSAT, the response is written into a buffer and the buffer is placed at the HEAD of the downlink queue associated with the BSAT. The buffer-header timestamp field is set up so as to expedite the transmission of the response to the BSAT. If local control packets change the Simulator's parameters for the BSAT, or if they reset or loop the Simulator/BSAT interface, such changes will be reflected in the common memory segment.

If an uplink process is in a looped state as a result of a LOOPBACK local control packet, burst packets are flagged in a header word as having been looped and are placed directly on the associated downlink queue. The local ESI transmission times of the packets are checked for validity. There are no other modifications/operations on the packet. Local control packets will be processed normally when an uplink process is looped.

An uplink process must map in all of the buffer pools for the processor node on which its associated receive-only I/O driver is located.

The functions performed by an uplink process could conceivably be incorporated as part of the "user" subroutine invoked by the receive-only I/O driver process. A separate process will be used, however, since the uplink functions are in a sense "higher level" than the functions needed to manage the

associated I/O hardware. (This is analagous to the distinction in the BSAT between HAP-level and HostIO-level processes.) The same distinction is made on the downlink side of the Simulator, which is discussed below. Such process separation may simplify the implementation of future uplink and downlink Simulator functions, such as the insertion of satellite channel noise.

2.5.3.3 Downlink Processes

There will be one downlink process for each BSAT attached to the Simulator. A downlink process communicates with either one or two transmit-only I/O drivers, depending on whether its associated BSAT is connected to the Simulator by one or two physical links, respectively. The interface to each I/O driver consists of a separate pair of dual queues, as previously described. In addition, each downlink process communicates with:

a. the merge process and one or two uplink processes via a dual queue containing local control packets, looped burst packets, and non-looped burst packets destined for the given BSAT,

b. the same (one or two) uplink processes via a common memory segment.

A downlink process passes looped burst packets found on its queue to a transmit-only I/O driver without making any modifications to the packets' contents. When a non-looped burst

packet is received (from the merge process), loop and reset indicators located in the common memory segment are checked. If the BSAT is looping or resetting the Simulator, all of these packets are dropped, effectively filtering out the BSAT's satellite downlink.

A downlink process assumes that the non-looped burst packets found on its downlink queue have been merged: They are found on the queue in satellite-time order with their satellite times in the local transmit time field and (possibly) corrupted bits in their BSAT CONTROL INFO or OPTIONAL MSGS areas.

When the Simulator is not being looped or reset by the BSAT, the merged burst packets are processed as follows: The channel symbol rate and the state-1 modulation type and coding rate of the packet are compared with the current values of these parameters for the receiving BSAT (found in the common memory segment). A packet with a mismatch is dropped. The local transmit time field is then incremented by 1/2 RTT for the downlink BSAT (plus the burst's preamble size), to provide the BSAT with the correct local receive time. The local receive time is then used to calculate a BIO transmit time, which is placed in the buffer header time-stamp field for use by an I/O driver. The packet is then placed on a transmit-request queue for processing by an I/O driver.

A downlink process performs (minimal) processing on any received local control packets and passes the results to an I/O driver.

When there are two transmit-only I/O drivers per BSAT, it is important that they each receive about 1/2 of the bandwidth of data being transmitted to the BSAT by the downlink process. The downlink process will attempt to equalize the amount of data sent to the two I/O drivers by using the total data size field provided with each packet. This can simply be done by maintaining the difference between the number of bytes sent to driver #1 and driver #2 as a signed integer. If the number is positive, the next packet will be sent to driver #2, and its size will be subtracted from the number; if negative, the packet will be sent to driver #1 and an addition will be done. This scheme is better than equally dividing the number of PACKETS sent to the I/O drivers, since it weights the packets by their quantity of transmit data.

Any packet that is to be sent to an I/O driver by a downlink process must be copied into physical memory on the processor node on which the I/O driver (and the corresponding I/O port) is located. Also, the downlink process cannot write into the original copy of the packet, since it may be shared with other downlink processes. In order to perform all of the possible copy

operations, a downlink process must map in all of the buffer pools on any processor node with a receive-only I/O port, as well as all of the buffer pools on the processor node(s) hosting the transmit-only port(s) for the associated BSAT.

A downlink process will not (except for processing/scheduling latencies) delay any of the packets it receives. Any explicit delay, beyond that already suffered by the packets while waiting on an uplink queue, will occur while waiting on a synchronous transmitter's queue.

2.5.3.4 Merge (Satellite) Process

The merge process (Merge) simulates the broadcast and multiaccess functions of a satellite. As such, it tries to remain as independent as possible of the state of the remainder of the Simulator. (For example, the merge process has no knowledge of the fact that a given BSAT interface is in a looped state.) The interfaces to the merge process are the uplink queues, each containing the non-looped burst packets from a single uplink process, and the downlink queues, each identically containing the results of the satellite merge for transmission to a single downlink process.

The merge process does not associate the (two) uplink queues that may contain packets from the same BSAT; it only expects that the satellite times found in the local transmit time field of the packets in a given queue are in increasing order. Under this assumption, Merge always processes the packet at the head of a queue before inspecting the contents of any following packets.

For the packet at the head of each uplink queue, Merge calculates the {T1,T2,T3,T4} values describing the sections of the corresponding burst. The T1 values are then used to sort the bursts at the heads of the uplink queues by increasing PREAMBLE transmission time. Merge then determines a decision time, and dismisses until that time has passed or more burst packets arrive (on any uplink queues that had previously been empty). The decision time is the real time indicated in the buffer header timestamp field (1/2 RTT + K, inserted by uplink process) of the packet at the head of the sorted list. When any new packets arrive on previously empty uplink queues, the sorted list and decision time is updated and Merge again dismisses.

When the decision time has passed, a merge decision is made using the model described in "Burst Merger and Reception" above. Each decision has one of five possible results:

a. The packet at the head of the list sorted by T1 values is passed unscathed to the downlink,

b. the head packet is passed to the downlink with its OPTIONAL MSGS area corrupted and the next packet on the list is dropped,

c. the head packet is passed to the downlink with its BSAT CONTROL INFO area corrupted and the next packet is dropped,

d. the head packet and the next packet are both dropped, or

e. only the head packet is dropped.

Merge advances a satellite time pointer every time it makes a new decision. This pointer defines the satellite time up to which decisions have been made. It is also used to detect packets that arrive either out of order or too late on their uplink queues; such packets will be dropped. For example, if Merge makes a decision with an outcome of type "a", the satellite time pointer would be updated to be equal to the T4 value of the burst packet being passed to the downlink. (T4 defines the satellite time of the end of the burst.) This is done to reflect Merge's decision that the burst has not suffered a collision with any other burst at the satellite. If a burst packet were then to arrive on an uplink queue with a T1 value (defining the beginning of the burst) that is earlier than the satellite time pointer, it would be dropped. The packet is dropped because (1) its T1 value may be such that it would invalidate the (irrevocable) decision that the previous packet suffered no collisions, and (2) its T1 value may be such that the packet would be passed to the downlink out of satellite time order.

The value of K used to adjust Merge's decision times must be large enough to account for any latencies suffered by a packet before reaching its uplink queue. If the value is too small, packets may be unfairly dropped by Merge for tardiness.

In order to perform its functions, Merge must map in all of the buffer pools on every processor node with a receive-only I/O port. Merge passes a burst packet to the downlink by placing a pointer to the packet (the buffer ID for the packet) on each of the downlink queues. The downlink processes will then make their own private copies of the packet.

2.5.3.5 System Initialization/Control Process

The BSAT Satellite Simulator will have a single initialization/control process (Init). The initialization scheme to be used for the Simulator is similar to that used for the BSATs. Init will be loaded into a Butterfly Multiprocessor node, and will be invoked via the Chrysalis operating system's user interface.

When Init starts to run it will allocate and map in a memory segment referred to as the "common segment." The common segment will be mapped in by all of the Simulator processes and will be used as a means of interprocess communication. Most common

segment data entries will be single words or bytes defined as write-only by one process and read-only by other processes; this avoids waiting on locks for common segment accesses. Some of the common segment contents are indicated below.

Init will perform TTY I/O with the Simulator user in order to determine the desired Simulator configuration/parameters (e.g., how many BSATs are to be supported, which nodes the various Simulator processes are to operate on). The configuration/parameters chosen will be entered in the common segment. Init will then make the appropriate calls to Make_Template and Make_Process for each desired process in turn, passing the OID of the common segment and other configuration information as arguments 1 and 2 of each process invocation, respectively (e.g., for an uplink process, the BSAT number and physical link number are passed in argument 2). When each process runs, it will peform various initializations and will create assorted Chrysalis objects (primarily event blocks, dual queues, and channel control blocks, but NOT buffer pools). The object handles of the objects that are used for interprocess communication are published in the common segment. When initialization is complete, the process will signal Init via an event handle found in the common segment and dismisses; Init will then invoke the next process.

When the last process has signaled Init that its initialization is complete, Init creates buffer pools occupying most of the remaining memory on each processor node that has synchronous I/O ports. Again, the object handles are published in the common segment. Init will then signal each process in turn. When each process is so signaled it will use entries in the common segment to map in its required buffer pools, make local copies of interprocess queue handles, etc. The process will then enter its main loop and will signal Init that it has done so. The receive-only I/O driver processes will be the last processes to be started by Init; this is done to prevent packets from waiting on queues for service by a process that has not yet entered its run mode.

Once all of the processes have signaled Init that they are running, Init enters a control mode, where it again interacts with the Simulator user. In this case the user is given control of TTY display of Simulator statistics. It is expected that there will be many Simulator statistics kept in the common segment by the various processes. Init should provide a facility to take snapshots of these. The user will also be given mechanisms to alter various Simulator parameters, reset the Simulator, etc.

3 MOBILE ACCESS TERMINAL NETWORK

Our participation in the development of the Mobile Access Terminal (MAT) and the MAT Satellite Network (MATNET) during the last quarter was reduced to a low-level support while waiting for contract renewal. We did, however, participate in several meetings with Navelex to discuss the next-generation MATNET system.

DISTRIBUTION

ARPA
Director (3 copies)
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209
Attn: Program Manager
R. Kahn
R. Ohlander
B. Leiner

DEFENSE DOCUMENTATION CENTER (12 copies)
Cameron Station
Alexandria, VA 22314

DEFENSE COMMUNICATIONS ENGINEERING CENTER
1860 Wiehle Road
Reston, VA 22090
Attn: Maj. J. Fredricks

DEPARTMENT OF DEFENSE
9800 Savage Road
Ft. Meade, MD 20755
Attn: R. McFarland C132 (2 copies)

DEFENSE COMMUNICATIONS AGENCY
8th and South Courthouse Road
Arlington, VA 22204
Attn: Code B645
Glynn Parker, Code B626

NAVAL ELECTRONIC SYSTEMS COMMAND
Department of the Navy
Washington, DC 20360
Attn: B. Hughes, Code 6111
F. Deckelman, Code 6131

MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02138
Attn: D. Clark

MIT Lincoln Laboratory
244 Woods Street
Lexington, MA 02173
Attn: C. Weinstein

DISTRIBUTION cont'd

USC Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 90291
Attn: D. Cohen
S. Casner

DISTRIBUTION cont'd

BOLT BERANEK AND NEWMAN INC.
1300 North 17th Street
Arlington, VA 22209
Attn: E. Wolf

BOLT BERANEK AND NEWMAN INC.
10 Moulton Street
Cambridge, MA 02238

S. Blumenthal
M. Brescia
R. Bressler
J. Byrd
P. Cudhea
A. Echenique
R. Edmiston
W. Edmond
L. Evenchik
G. Falk
J. Goodhue
S. Groff
R. Gurwitz
J. Haverty
F. Heart
J. Herman
R. Hinden
D. Hunt
S. Kent
A. McKenzie
D. Melone
W. Milliken
R. Newman
M. Nodine
R. Rettberg
H. Rising
J. Robinson
E. Rosen
P. Santos
J. Sax
S. Storch
R. Thomas
B. Woznick
Library